Advanced Programming

Data Structures in C++

Dr. Zeyad Safaa Younus Saffawi

Introduction

- In C/C++, **arrays** allow us to store multiple values of the **same data type**.
- Example:

```
int numbers[5];
```

- All elements must be **integers**.
- However, sometimes we need to store **different types of data together**. For example, information about a **book** may include:
  - Title
  - Author
  - Subject
  - Book ID
- In this case, we use a **Structure**.
- A **structure (struct)** is a **user-defined data type** that allows combining **different types of data in one variable**.

Defining a Structure

- To define a structure, we use the keyword **struct**.
- **General Syntax**

```
struct StructureName
{
    dataType member1;
    dataType member2;
    dataType member3;
};
```

- **Example: Book Structure**

```
struct Books
{
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
};
```

This structure contains four members:

Member	Type
title	char array
author	char array
subject	char array
book_id	integer

Declaring Structure Variables

- After defining the structure, we can create variables of that type.

- **General Syntax**

```
struct StructureName Structurevariable;
```

Example:

```
struct Books Book1;
struct Books Book2;
```

- Now **Book1** and **Book2** are variables of type **Books**.

Accessing Structure Members

- To access structure members, we use the **dot operator (.)**

Syntax

```
structureVariable.member;
```

Example

```
Book1.book_id = 1234;
```

Example: Write a C++ program to define a structure called Books containing the following members:

- title (**char**[])
- author (**char**[])
- book_id (integer)
- subject (**char**[])

Then create one object of the structure, assign values to its members, and display all the information on the screen.

```
#include <iostream>
#include <cstring>
using namespace std;
struct Books
{
   char title[50];
   char author[50];
   char subject[100];
   int book_id;
};
int main()
{
   struct Books Book1;
   strcpy(Book1.title, "Learn C++ Programming");
   strcpy(Book1.author, "Chand Miyan");
   strcpy(Book1.subject, "C++ Programming");
   Book1.book_id = 6495407;
   cout << "Book title: " << Book1.title << endl;
   cout << "Book author: " << Book1.author << endl;
   cout << "Book subject: " << Book1.subject << endl;
   cout << "Book ID: " << Book1.book_id << endl;
   return 0;
}
```

Example: Write a C++ program to define a structure called Books containing the following members:

- title (**string**)
- author (**string**)
- book_id (integer)
- subject (**string**)

Then create one object of the structure, assign values to its members, and display all the information on the screen.

```
#include <iostream>
using namespace std;
struct Books
{
   string title;
   string author;
   string subject;
   int book_id;
};
int main()
{
   Books Book1;
   Book1.title = "Learn C++ Programming";
   Book1.author = "Chand Miyan";
   Book1.subject = "C++ Programming";
   Book1.book_id = 6495407;
   cout << "Book title: " << Book1.title << endl;
   cout << "Book author: " << Book1.author << endl;
   cout << "Book subject: " << Book1.subject << endl;
   cout << "Book ID: " << Book1.book_id << endl;
   return 0;
}
```

Structures as Function Arguments

- A **structure** can be passed to a **function** just like any other variable.
- **Example**

```
void printBook (struct Books book)
{
   cout << "Book title: " << book.title << endl;
   cout << "Book author: " << book.author << endl;
   cout << "Book subject: " << book.subject << endl;
   cout << "Book ID: " << book.book_id << endl;
}
```

- **Calling the function:**

```
printBook(Book1);
```

- The structure **Book1** is passed to the function.

Pointers to Structures

- We can also create **pointers to structures**.

- **Declaration**

```
struct Books *ptr;
```

Assigning the address:

```
ptr = &Book1;
```

- **Accessing Members Using Pointer**
- When using a pointer, we use the **arrow operator (->)** instead of the dot operator.

Example

```
ptr->title
ptr->author
ptr->book_id
```

Example using Pointer

```
#include <iostream>
#include <cstring>
using namespace std;
struct Books
{
   char title[50];
   char author[50];
   char subject[100];
   int book_id;
};
void printBook(struct Books *book)
{
   cout << "Book title: " << book->title << endl;
   cout << "Book author: " << book->author <<
endl;
   cout << "Book subject: " << book->subject <<
endl;
   cout << "Book ID: " << book->book_id << endl;
}
int main()
{
   struct Books Book1;
   strcpy(Book1.title, "Learn C++ Programming");
   strcpy(Book1.author, "Chand Miyan");
   strcpy(Book1.subject, "C++ Programming");
   Book1.book_id = 6495407;
   printBook(&Book1);
   return 0;
}
```

The typedef Keyword

- The **typedef** keyword allows us to create a **new name (alias) for a data type**.
- Example:

```
typedef struct
{
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
} Books;
```

Now we can declare variables without writing **struct**.
Example:

```
Books Book1, Book2;
```

Example for typedef with Pointers

```
typedef long int *pint32;

pint32 x, y, z;
```

Here: **x, y, and z** are **pointers to long integers**.

Important concepts:

- Structures in C++ are used to:
  - Combine **different data types** in one variable
  - Represent real-world entities (student, book, employee)

Concept	Description
struct	define structure
.	access member
->	access member using pointer
typedef	create alias for data type

Example: Student Information

Write a program to define a structure called **Student** containing:

name (char[ ])

id

marks

Then read the data of one student and print it.

```
#include <iostream>
using namespace std;
struct Student
{
    char name[30];
    int id;
    float marks;
};
int main()
{
    Student s;
    cout << "Enter student name: ";
    cin.getline(s.name, 30);
    cout << "Enter student id: ";
    cin >> s.id;
    cout << "Enter student marks: ";
    cin >> s.marks;
    cout << "\nStudent Information\n";
    cout << "Name: " << s.name << endl;
    cout << "ID: " << s.id << endl;
    cout << "Marks: " << s.marks << endl;
    return 0;
}
```

Example: Student Information

Write a program to define a structure called **Student** containing:

name (string)
id
marks

Then read the data of one student and print it.

```
#include <iostream>
using namespace std;
struct Student
{
    string name;
    int id;
    float marks;
};
int main()
{
    Student s;
    cout << "Enter student name: ";
    getline(cin, s.name);
    cout << "Enter student id: ";
    cin >> s.id;
    cout << "Enter student marks: ";
    cin >> s.marks;
    cout << "\nStudent Information\n";
    cout << "Name: " << s.name << endl;
    cout << "ID: " << s.id << endl;
    cout << "Marks: " << s.marks << endl;
    return 0;
}
```

Example : Employee Salary

Write a program to define a structure called **Employee** containing:

- name (**char [ ]**)
- id
- salary

Then store data for **two employees** and print their information.

```
#include <iostream>
using namespace std;
struct Employee
{
   char name[30];
   int id;
   float salary;
};

int main()
{
   Employee e1, e2;
   cout << "Enter first employee name: ";
   cin.ignore();
   cin.getline(e1.name, 30);
   cout << "Enter id: ";
   cin >> e1.id;
   cout << "Enter salary: ";
   cin >> e1.salary;
   cin.ignore();
    cout << "Enter second employee name: ";
   cin.getline(e2.name, 30);
   cout << "Enter id: ";
   cin >> e2.id;
   cout << "Enter salary: ";
   cin >> e2.salary;
   cout << "\nEmployee 1\n";
   cout << e1.name << " " << e1.id << " " << e1.salary << endl;
   cout << "\nEmployee 2\n";
   cout << e2.name << " " << e2.id << " " << e2.salary << endl;
   return 0;
}
```

Example : Employee Salary

Write a program to define a structure called **Employee** containing:

- name (**string**)
- id
- salary

Then store data for **two employees** and print their information.

```
#include <iostream>
using namespace std;
struct Employee
{
   string name;
   int id;
   float salary;
};
int main()
{
   Employee e1, e2;
   cout << "Enter first employee name: ";
   getline(cin, e1.name);
   cout << "Enter id: ";
   cin >> e1.id;
   cout << "Enter salary: ";
   cin >> e1.salary;
   cin.ignore();
   cout << "Enter second employee name: ";
   getline(cin, e2.name);
   cout << "Enter id: ";
   cin >> e2.id;
   cout << "Enter salary: ";
   cin >> e2.salary;
   cout << "\nEmployee 1\n";
   cout << e1.name << " " << e1.id << " " << e1.salary << endl;
   cout << "\nEmployee 2\n";
   cout << e2.name << " " << e2.id << " " << e2.salary << endl;
   return 0;
}
```

Example : Employee Salary

Write a program to define a structure called **Employee** containing:

- name (**char[ ]**)
- id
- salary

Then store data for **Five employees** and print their information.

```
#include <iostream>
#include <limits>
using namespace std;
struct Employee
{
   char name[30];
   int id;
   float salary;
};
int main()
{
   Employee e[5];
   for (int i = 0; i < 5; i++)
   {
      cout << "\nEnter employee " << i + 1 << " name: ";
      cin.ignore(numeric_limits<streamsize>::max(), '\n');
      cin.getline(e[i].name, 30);
      cout << "Enter id: ";
      cin >> e[i].id;
      cout << "Enter salary: ";
      cin >> e[i].salary;
   }
   cout << "\n===== Employees Data =====\n";
   for (int i = 0; i < 5; i++)
   {
      cout << "\nEmployee " << i + 1 << endl;
      cout << "Name: " << e[i].name << endl;
      cout << "ID: " << e[i].id << endl;
      cout << "Salary: " << e[i].salary << endl;
   }
   return 0;
}
```

Example : Employee Salary
Write a program to define a structure called **Employee** containing:
name (using **string**)
id
salary
Then store data for **Five employees** and print their information.

```
#include <iostream>
using namespace std;
struct Employee
{
   string name;
   int id;
   float salary;
};
int main()
{
   Employee e[5];
   for (int i = 0; i < 5; i++)
   {
      cout << "\nEnter employee " << i + 1 << " name: ";
      getline(cin, e[i].name);
      cout << "Enter id: ";
      cin >> e[i].id;
      cout << "Enter salary: ";
      cin >> e[i].salary;
      cin.ignore();
 }
    cout << "\n===== Employees Data =====\n";
   for (int i = 0; i < 5; i++)
   {
      cout << "\nEmployee " << i + 1 << endl;
      cout << "Name: " << e[i].name << endl;
      cout << "ID: " << e[i].id << endl;
      cout << "Salary: " << e[i].salary << endl;
   }
   return 0;
}
```

Example : Using Pointer with Structure:

Write a program to define a structure called **Book** and use a **pointer to structure** to print the data.

Solution

```
#include <iostream>
using namespace std;
struct Book
{
   char title[30];
   int id;
};
int main()
{
   Book b;
   Book *ptr;
   ptr = &b;
   cout << "Enter book title: ";
   cin.getline(b.title,30);
   cout << "Enter book id: ";
   cin >> b.id;
   cout << "\nBook Information\n";
   cout << "Title: " << ptr->title << endl;
   cout << "ID: " << ptr->id << endl;
   return 0;
}
```

References

- Gaddis, T. (2014). *Starting out with C++: From control structures through objects* (8th ed.). Pearson.
- Soulié, J. (2007, April 24). C++ language tutorial. cplusplus.com.
- Tutorials Point. (n.d.). *Learn C++ programming language*. Tutorials Point.